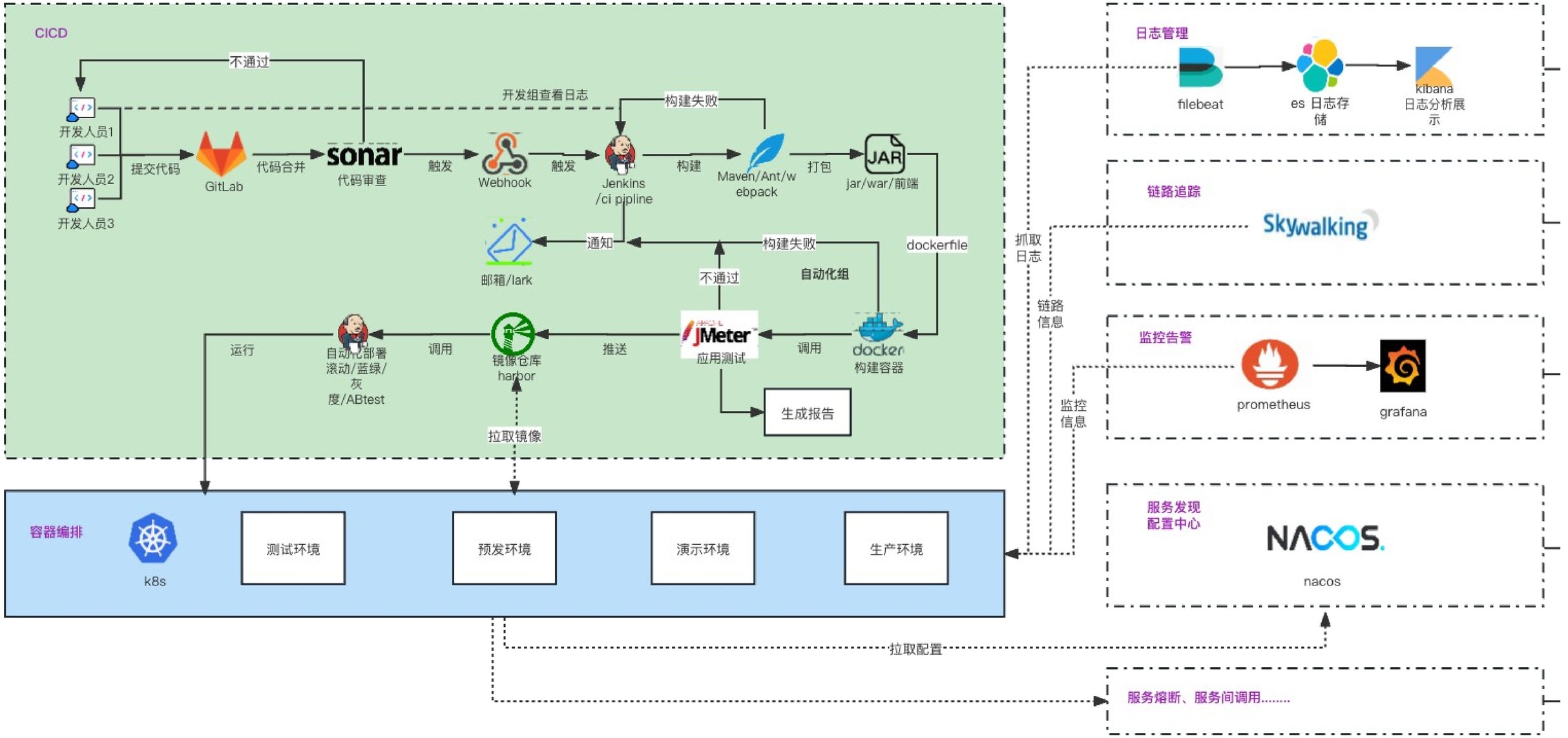
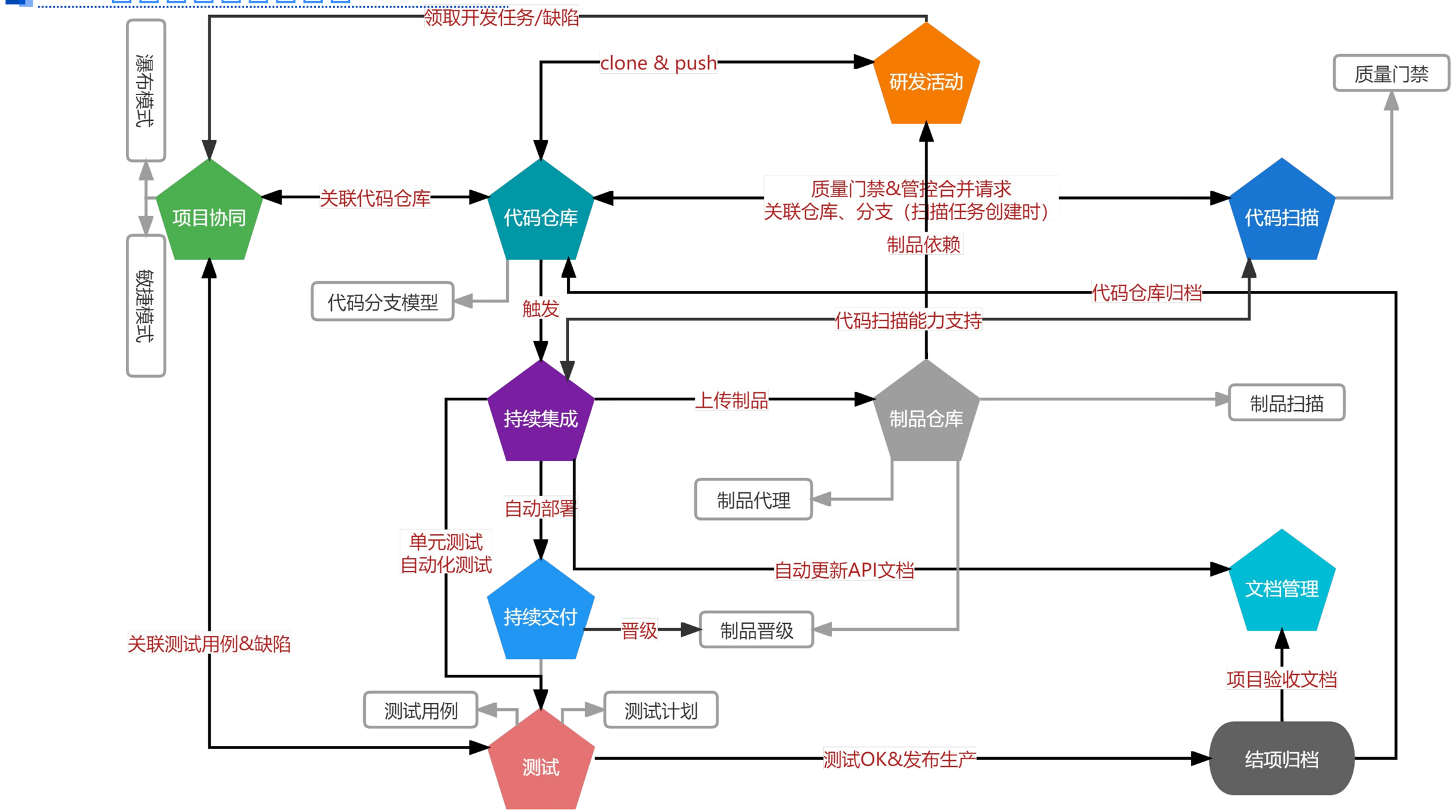
Devops □□

DevOps □□□□

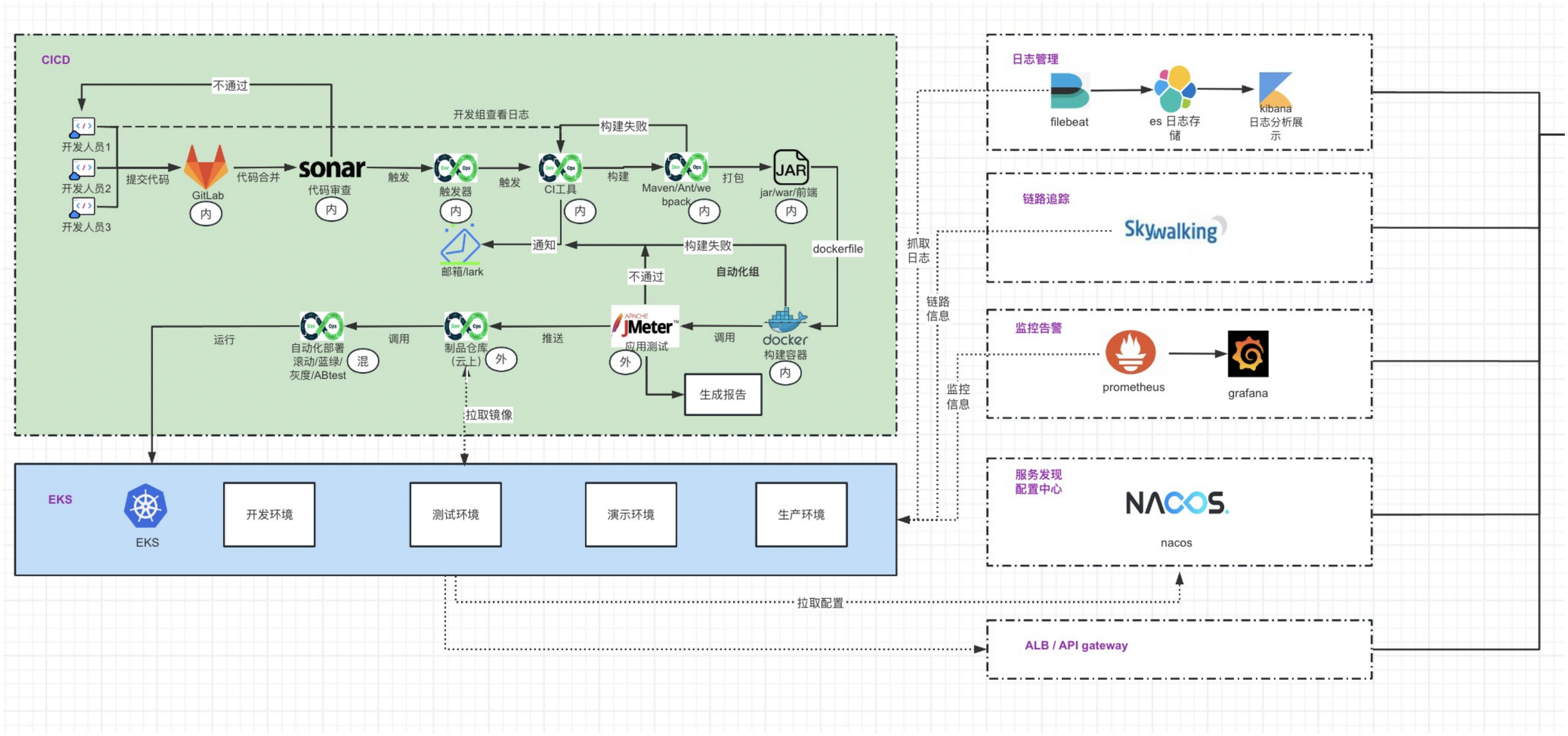
Gartner □ DevOps □□□□□□ IT □□□□□□

□□□□□□□□ - □□


CICD
不通过
开发人员1
开发人员2
开发人员3
提交代码
GitLab
代码合并
sonar
代码审查
触发
Webhook
触发
Jenkins
/ci pipline
构建
Maven/Ant/w
ebpack
打包
JAR
jar/war/前端
开发组查看日志
构建失败
通知
邮箱/lark
构建失败
dockerfile
不通过
自动化组
运行
自动化部署
滚动/蓝绿/
灰
度/ABtest
调用
镜像仓库
harbor
推送
JMeter
应用测试
调用
docker
构建容器
生成报告
拉取镜像
容器编排
k8s
测试环境
预发环境
演示环境
生产环境
日志管理
filebeat
es 日志存
储
kibana
日志分析展
示
抓取
日志
链路追踪
Skywalking
链路
信息
监控告警
prometheus
grafana
监控
信息
服务发现
配置中心
NACOS.
nacos
拉取配置
服务熔断、服务间调用.....

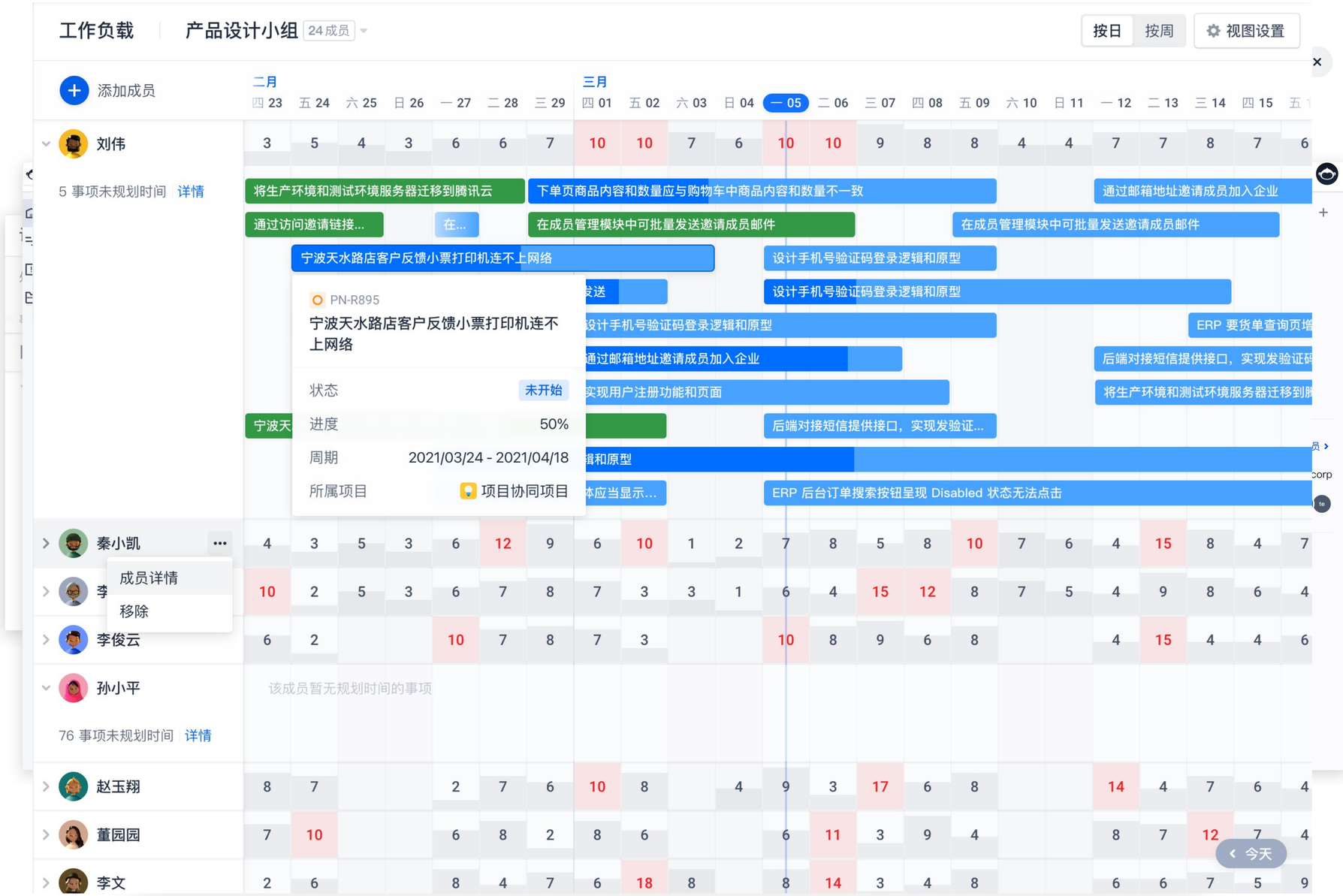

瀑布模式
敏捷模式
项目协同
领取开发任务/缺陷
clone & push
研发活动
质量门禁
关联代码仓库
代码仓库
质量门禁&管控合并请求
关联仓库、分支（扫描任务创建时）
代码扫描
制品依赖
代码分支模型
触发
代码仓库归档
代码扫描能力支持
持续集成
上传制品
制品仓库
制品扫描
制品代理
自动部署
单元测试
自动化测试
自动更新API文档
文档管理
持续交付
晋级
制品晋级
关联测试用例&缺陷
项目验收文档
测试用例
测试计划
测试
测试OK&发布生产
结项归档

□□□□□□□□ - □□


CICD
不通过
开发人员1
开发人员2
开发人员3
提交代码
GitLab
内
代码合并
sonar
代码审查
内
触发
开发组查看日志
构建失败
触发器
内
触发
CI工具
内
构建
Maven/Ant/webpack
内
打包
JAR
jar/war/前端
内
dockerfile
通知
构建失败
邮箱/lark
不通过
自动化组
运行
自动化部署
滚动/蓝绿/
灰度/ABtest
混
调用
制品仓库
（云上）
外
推送
JMeter
应用测试
外
调用
docker
构建容器
内
生成报告
拉取镜像
EKS
EKS
开发环境
测试环境
演示环境
生产环境
日志管理
filebeat
es 日志存储
kibana
日志分析展示
抓取日志
链路追踪
Skywalking
链路信息
监控告警
prometheus
grafana
监控信息
服务发现
配置中心
NACOS
nacos
拉取配置
ALB / API gateway

组织转型

组织
流程
能效
工具
文化
需求负责人
开发负责人
教练
产品
开发
测试分析
教练
开发团队
运维团队
测试团队
运维团队
维护负责人
原始需求
项目管理
需求分析
产品需求池
敏捷迭代
迭代 / 史诗管理
需求 / 任务管理
缺陷管理
传统迭代
瀑布管理
任务分派
代码提交
持续集成
开发环境
质量合规
质量不合规
持续迭代
代码提交 /Review
持续集成
Dev & Test 环境
集成测试
自动化测试
制品进阶
SIT 环境
持续交付
持续反馈
UAT 环境
生产环境
过程数据采集
数据度量
数据分析
过程改进
需求评审 / 流转规范
分支管理规范
版本管理规范
开发 / 测试过程管理规范 + 经验总结
制品管理规范
交付与运维规范
持续交付
需求 / 开发管理
原始需求
项目管理
审批管理
协作管理
工作量管理
计划管理
软件架构
微服务
容器化
开发脚手架
敏捷协同 + 代码托管 + CI/CD + 制品管理
迭代管理
需求管理
代码管理
流水线管理
应用管理
制品管理
任务管理
缺陷管理
分支管理
构建环境
部署流程
版本管理
敏捷看板
流程管理
研发规范
质量红线
变更控制
插件整合
自动化测试
接口测试
UI 测试
APP 测试
度量管理
度量仪表盘
可视化插件
数据收集
PaaS 管理平台
容量管理
运维看板
中间件管理
CMDB
全景监控
日志分析
信任
协作
自组织
自管理
价值交付
持续改进
可视化度量
学习型组织



项目集管理特性

业务需求流转与分解

- 多项目完成共同的业务目标
- 业务目标里程碑管理
- 项目集概览与风险管理
- 事务分解到研发项目中
- 工作负载管理

项目集管理特性

Scrum 敏捷研发管理

- 迭代管理与规划
- 自定义泳道与每日看板
- 事务流转与自动化
- 事务的分解与多重关联

企业级代码仓库

Git & SVN 多仓库支持

支持创建与管理多种仓库类型，通过企业级的代码托管方案，使用统一的视角管理企业内部的代码资产，支持外部代码库导入。

分支 & 版本 & 标签管理

支持代码分支管理，设置保护分支，以及指定分支管理员，分支管理员控制代码合并准入，支持版本及标签管理。

多维度的安全访问管控

支持基于用户组、仓库设置访问及操作权限，也支持文件及目录锁定、防暴力破解、访问者水印、**IP** 白名单、审计日志等，满足企业的安全管控要求。

代码提交关联事项

支持查看提交记录，支持提交记录和合并请求关联其它事项，比如关联需求、任务、缺陷，实现开发代码与工作事项关联。

研发流程管理

支持设置研发规范，通过研发规范可以约束分支命名规则、分支合流规则、版本管理规则等，帮助团队有效落地研发流程规范化管理。

代码扫描能力

支持代码质量扫描，可以支持目前常见的所有主流语言，内置接近上万种扫描规则，通过代码扫描对提交代码进行质量管控，支持质量门禁设置，严格控制代码准入质量。

代码评审机制

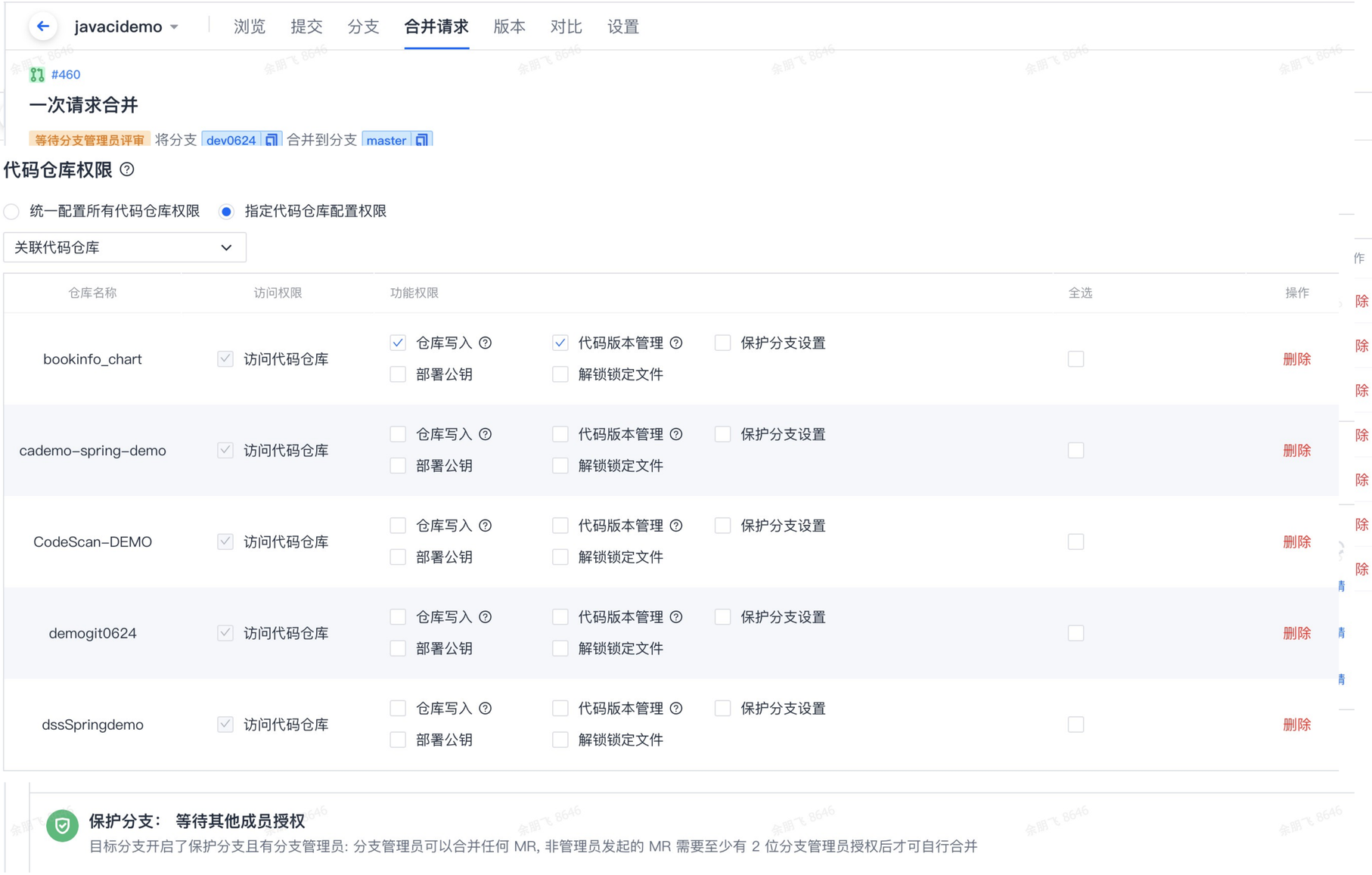
支持合并请求时指定评审人员，也支持合并请求允许合并授权数量设置，相关人员评审通过以后才可进行代码合并操作，评审时支持代码逐行对比、逐行评论，严控代码合入质量。

平台开放性强

平台提供丰富的 **OpenAPI** 接口以及 **WebHook** 集成方式，简单易用，可满足各个功能层面的第三方集成需求。

数据高可用存储及备份机制

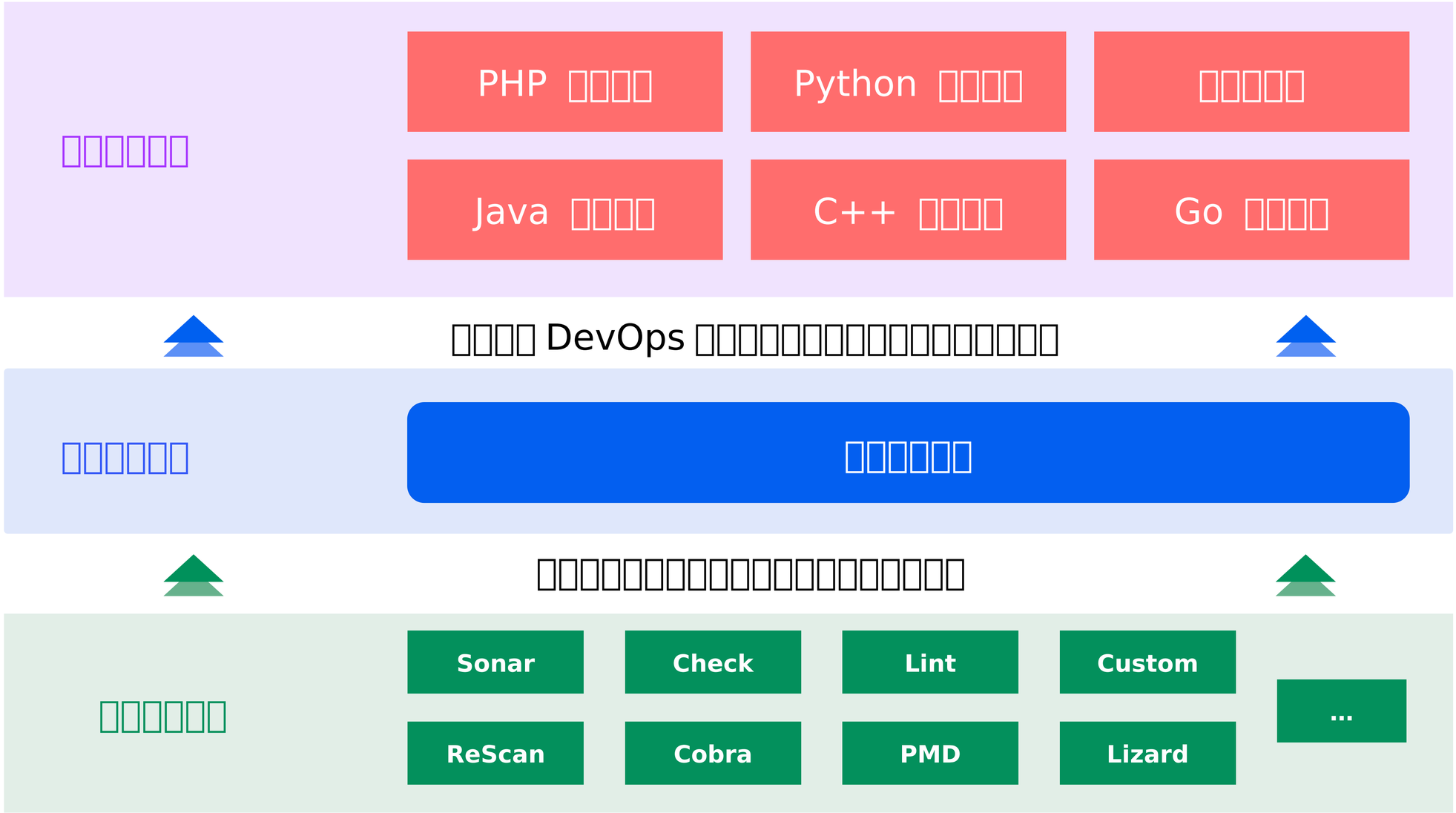
支持数据多副本分片存储，部分节点的宕机不会影响服务可用性和数据完整性，确保数据的高可用存储。同时，提供完备的数据备份、恢复方案，以多种方式确保数据安全稳定。



代码仓库特性

安全可控的代码管理

- 代码文件锁定
- Git 仓库目录级别权限控制
- 通过自动化辅助评审
- 精细化的权限管理



代码扫描

代码扫描方案
PHP 推荐方案
Python 推荐方案
自定义方案
Java 推荐方案
C++ 推荐方案
Go 推荐方案
依托丰富 DevOps 经验，输出各类技术栈的扫描方案实践
代码扫描平台
代码扫描平台
整合工具扫描能力，支持多语言与技术栈扫描
代码扫描工具
Sonar
Check
Lint
Custom
ReScan
Cobra
PMD
Lizard
...

代码扫描

质量左移及代码规范

- 自动匹配代码仓库语言
- 实时监控代码仓库质量
- 历史趋势展示与问题跟踪
- 多样化的问题处理方式
- 自定义工具及规则集成

□□□□

1 □□□□□□□□

□□ **Python** □ **Java** □ **Node.js** □ **Go** □ **.NET** □□□□□□□□

持续集成

个性化编排及自动化执行

- 图形化编排 & Pipeline As Code
- 丰富的插件市场
- 流水线模板，辅助业务标准化
- 池化构建资源，支持大并发
- 异构技术栈支持

制品统一管理

多样化的制品类型

支持管理多种主流的制品类型，包括 **Generic** 通用文件、**Docker Image**、**Maven**、**NPM**、**PyPi**、**Helm Chart**、**Composer**、**NuGet**、**Conan**、**CocoaPo**

丰富的元数据管理

能够自动解析制品中的元数据，帮助开发者更加直观的了解制品的信息。同时，也支持对于元数据的自定义。通过元数据的自定义，能够实现制品上下游信息的聚合及特殊业务属性的定义满足管理诉求

多数据中心下同步及分发

能够支持在多数据中心下进行多团队研发协作的制品同步及针对生产视角下具备生产发布标签的制品分发到数据中心里进行应用投产

精细化的权限管理

支持精细化的权限管理，针对公司级、团队级、项目级等三种不同权限颗粒，支持配置用户组 / 角色关联每个粒度的权限。

制品质量扫描

支持在不访问代码的情况下，扫描制品以及其元数据，发现制品中的安全风险。支持与持续集成无缝集成，确保制品的质量可靠。

可信软件源及依赖管理

支持制品代理功能，统一管理团队内的制品源，保障团队内使用的制品有来源追溯。支持制品依赖追踪，协助用户找到系统中关键制品的使用情况。

□□□□□□

- □□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□
- □□□□□□□
- □□□□□□□□

新建发布单

新建☆

python-flask-app:master-5c94356704eb8405036f459072d9e725a8e6b73d 详情

导出报告 重新扫描

python-flask-app:master-5c94356704eb8405036f459072d9e725a8e6b73d

用户 项目助手 通过制品更新触发了扫描

扫描状态 未通过质量红线

扫描版本 master-5c94356704e...

对比版本 master-a4cc5140f9c...

红线标准 总数 危急 ⩽ 0
且 总数 高危 ⩽ 0
且 新增 危急 ⩽ 0
且 新增 高危 ⩽ 0

构建记录 662852#4

持续时间 1 分钟 16 秒

开始时间 3 个月前

制品仓库 test

漏洞概览

漏洞数量统计图

479 漏洞数量

危急 0 高危 239 ▲ 5 中危 216 低危 24

漏洞列表统计

漏洞等级 全部 漏洞变化 全部 仅展示推荐优先关注漏洞 仅展示未被忽略漏洞 搜索 CVE 漏洞编号... 搜索所属依赖...

CVE 漏洞编号	漏洞等级	所属依赖	引入版本	修复版本	操作
CVE-2020-11655	高危	sqlite3	1.0-3.31.1	--	•••

SQLite 输入验证错误漏洞

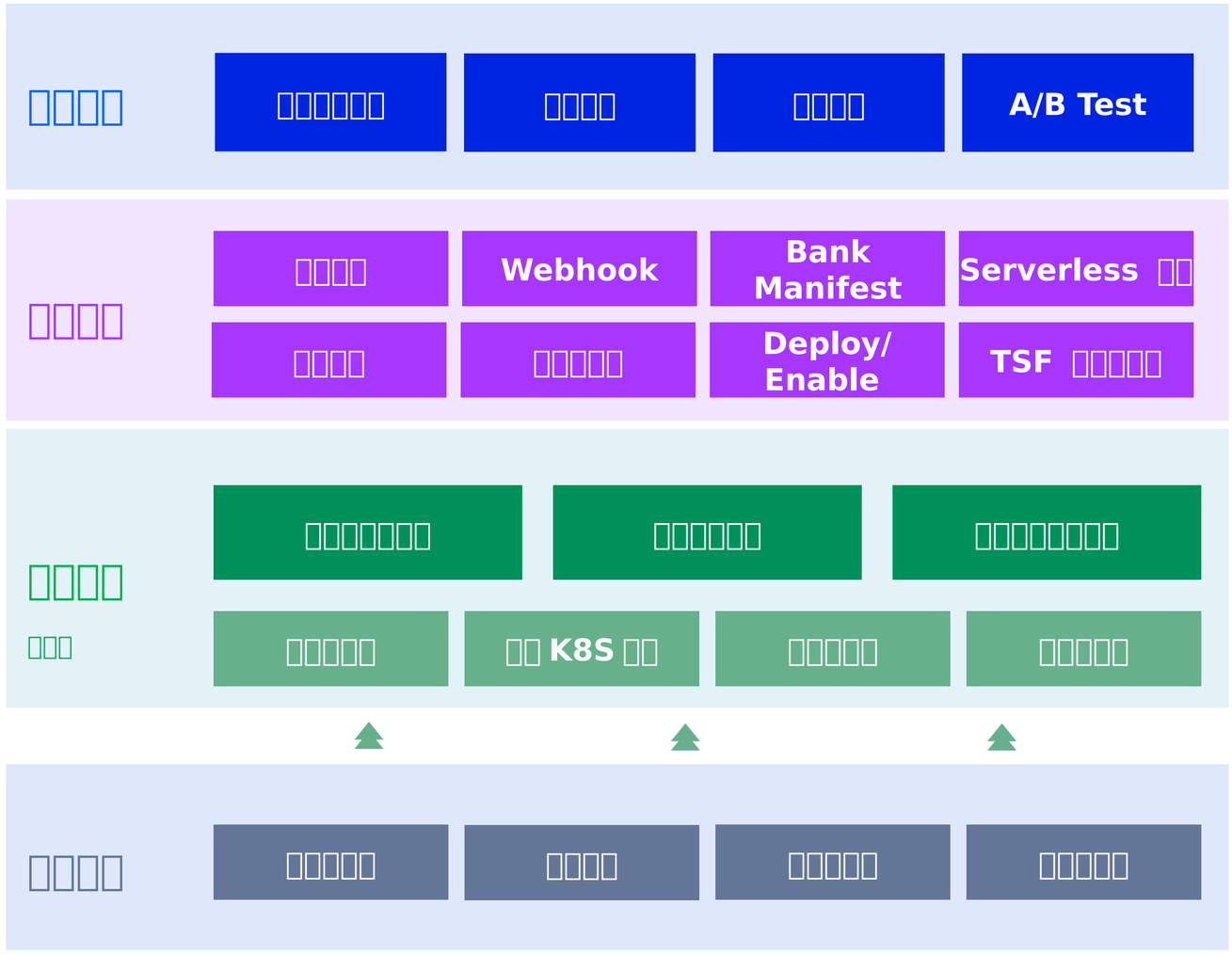
SQLite是美国D.RichardHipp软件开发者的一套基于C语言的开源嵌入式关系数据库管理系统。该系统具有独立性、隔离性、跨平台等特点。SQLite3.31.1及之前版本中存在安全漏洞，该漏洞源于程序没有正确处理AggInfo对象初始化。攻击者可利用该漏洞导致拒绝服务。

修复建议

目前厂商已发布升级补丁以修复漏洞，补丁获取链接：https://www3.sqlite.org/cgi/src/tktview?name=af4556bb5c

相关信息

https://lists.debian.org/debian-lts-announce/2020/05/msg00006.html
https://lists.debian.org/debian-lts-announce/2020/08/msg00037.html
https://security.FreeBSD.org/advisories/FreeBSD-SA-20:22.sqlite.asc
https://security.gentoo.org/glsa/202007-26
https://security.netapp.com/advisory/ntap-20200416-0001/
https://usn.ubuntu.com/4394-1/
https://www.oracle.com/security-alerts/cpujan2021.html



持续部署

部署场景
部署流程编排
灰度发布
蓝绿部署
A/B Test
部署步骤
主机部署
Webhook
Bank Manifest
Serverless 部署
流量调度
自定义步骤
Deploy/ Enable
TSF 微服务部署
部署通道
可扩展
基础资源透明化
灵活编排流程
兼容多种部署模式
虚拟机管道
标准 K8S 管道
公有云管道
定制化管道
资源管理
虚拟机环境
容器环境
公有云环境
混合云环境

TKE-BEIJING-PRO
production
触发器: 已启用
配置
立即
MANUAL START
余朋飞
4 个月 以前
00:44 00:33 00:34 00:39 00:07 - 00:40 00:43 00:23 14:18 00:37 00:10 00:38 00:39
00:18 00:18 00:13 00:10
状态: 成功
耗时: 18:11
查看制品 (15)
查看详情
预发布环境 - Migrate Database
预发布环境 - Deployment
是否灰度发布并 A/B 测试
生产环境 - Migrate Database
生产环境灰度 - Deployment
生产环境 - 灰度比例
生产环境 - 自动设置灰度
生产环境 - 是否全量发布
预发布环境 - Service
预发布环境 - Traefik
生产环境灰度 - Service
生产环境灰度 - AB Test Case
部署 (Manifest) 配置
基础设置
云账号
TKE-Beijing-pro
应用
gray-deploy-demo
覆盖命名空间
Manifest 配置
Manifest 来源
使用制品
输入内容
Manifest 制品
k8s-flask-dev-service
表达式计算
跳过 SpEL 表达式计算
需要绑定的制品
请选择制品

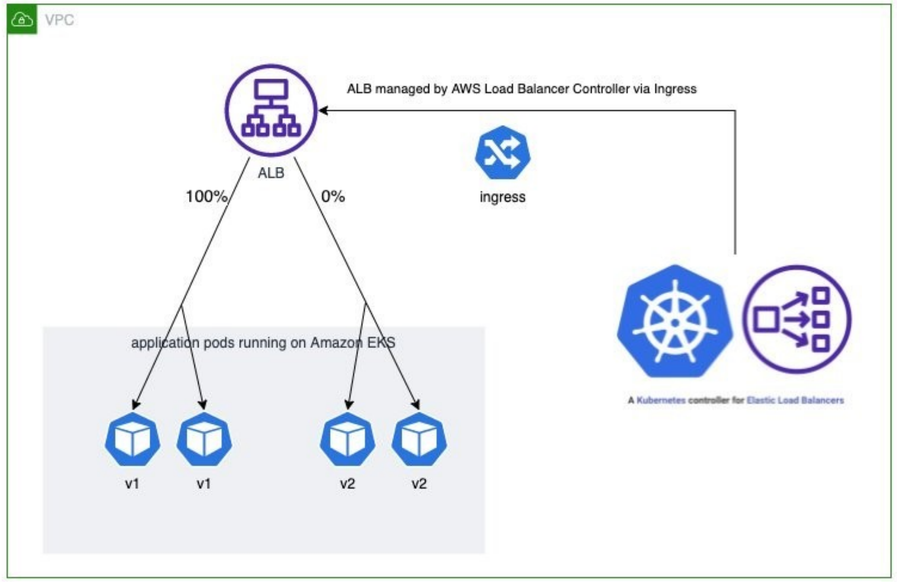
□□□□

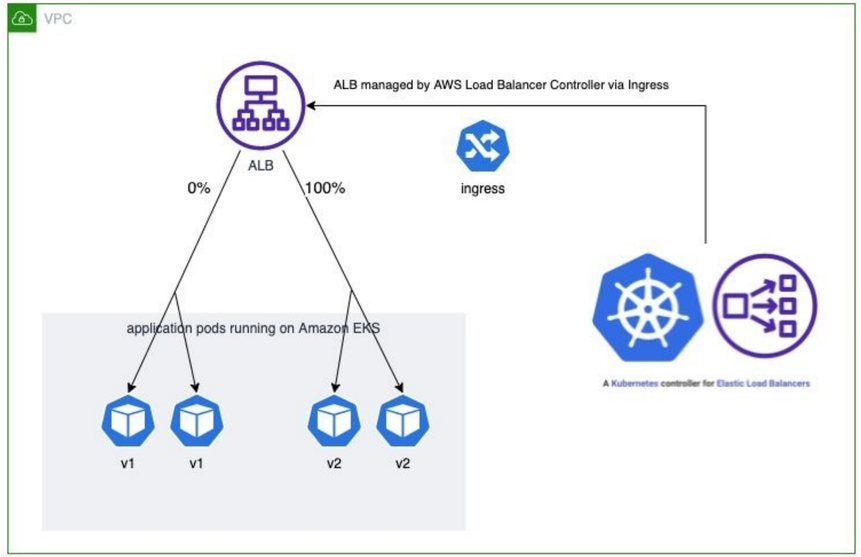
□□□□□□□□□□□□□□

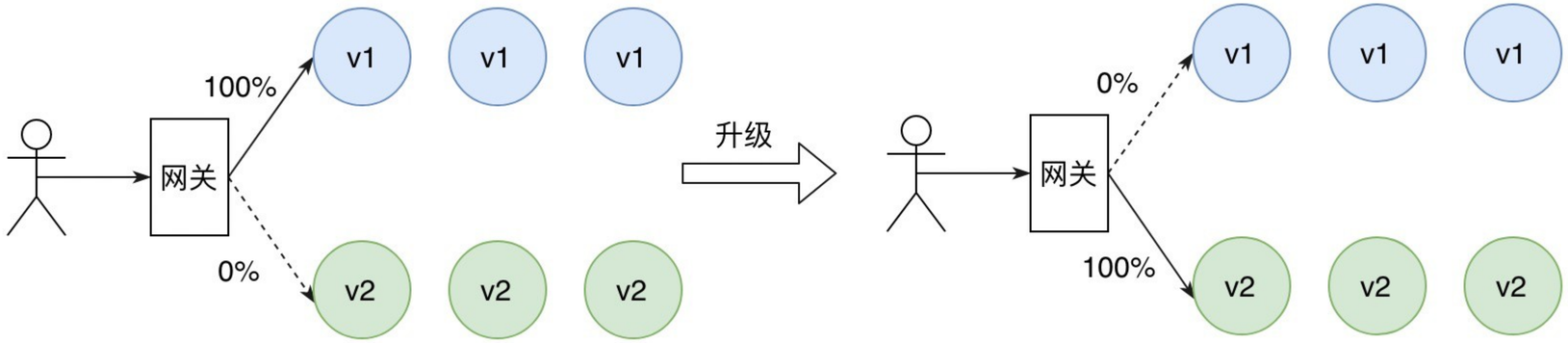
- □□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□

nginx-ingress
发布单
状态: 全部
创建人: 全部
部署流程: 全部
发布单状态
发布单名称
执行中
20210923-nginx-ingress-3. 灰
已终止
20210914-nginx-ingress-3. 灰
成功
20210914-nginx-ingress-3. 灰
已取消
k8s发布001
选择阶段
搜索阶段名称
依赖阶段: 基础配置
Kubernetes
主机组部署
TSF 部署
通用类型
部署 (Manifest)
部署 yaml/json 格式的 Kubernetes manifest 文件
选择
过滤 (Manifest)
过滤 (Manifest)
选择
扩缩容 (Manifest)
对 Kubernetes 对象执行扩缩容
选择
回滚 (Manifest)
回滚至目标版本
选择
删除 (Manifest)
删除 Kubernetes (Manifest)
选择
Bake (Manifest)
使用 Helm Bake manifest 文件
选择
Patch (Manifest)
Patch a Kubernetes object in place.
选择
应用
用初始化
3. 灰度发布流程
版本号
3.0.2
件路径
版本号

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: "hello-kubernetes"
  namespace: "hello-kubernetes"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-fa
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/actions.blue-green:
      {
        "type":"forward",
        "forwardConfig":{
          "targetGroups":[
            {
              "serviceName":"hello-kubernetes-v1",
              "servicePort":"80",
              "weight":100
            },
            {
              "serviceName":"hello-kubernetes-v2",
              "servicePort":"80",
              "weight":0
            }
          ]
        }
      }
  labels:
    app: hello-kubernetes
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: blue-green
                port:
                  name: use-annotation
```

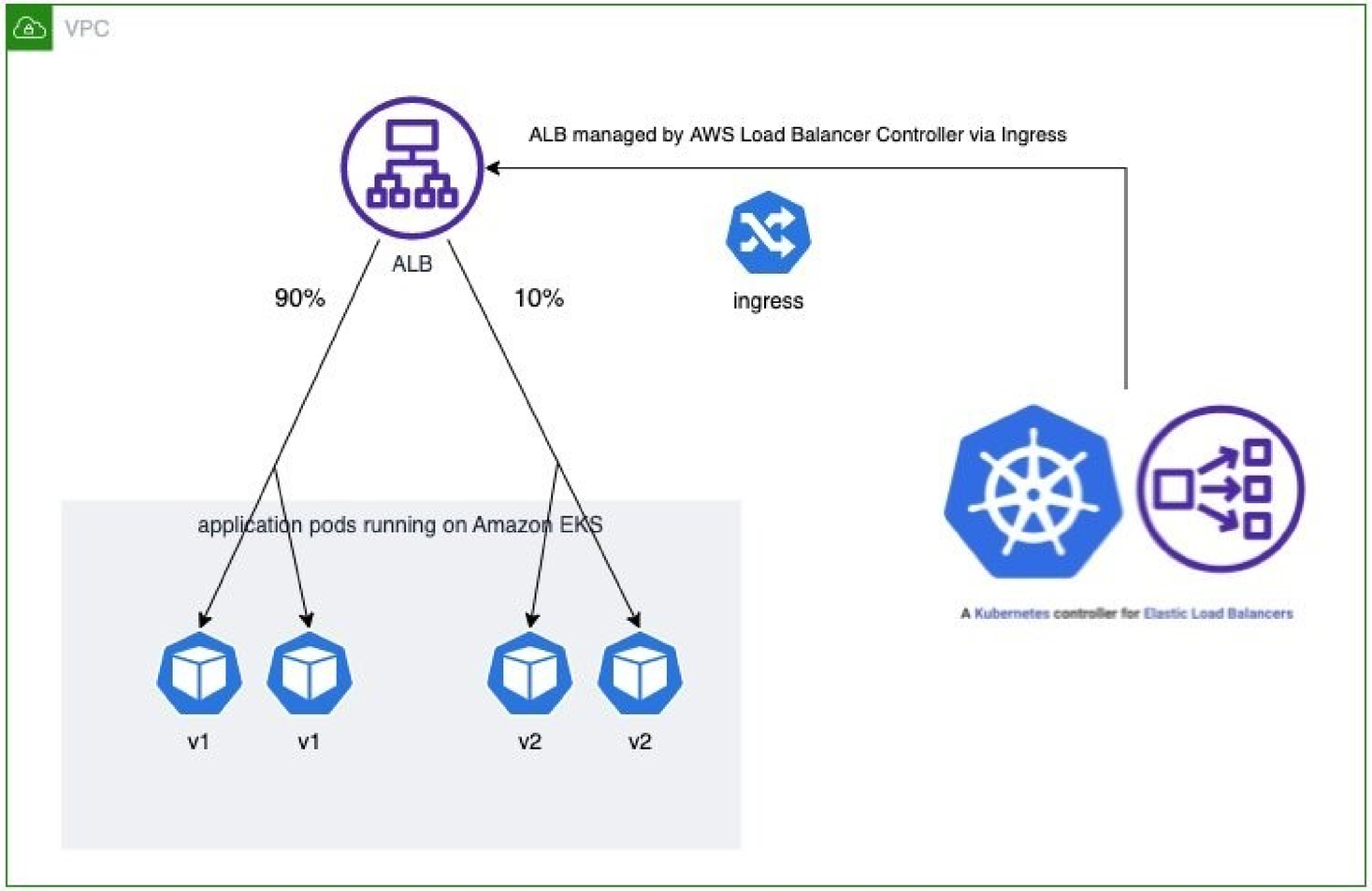






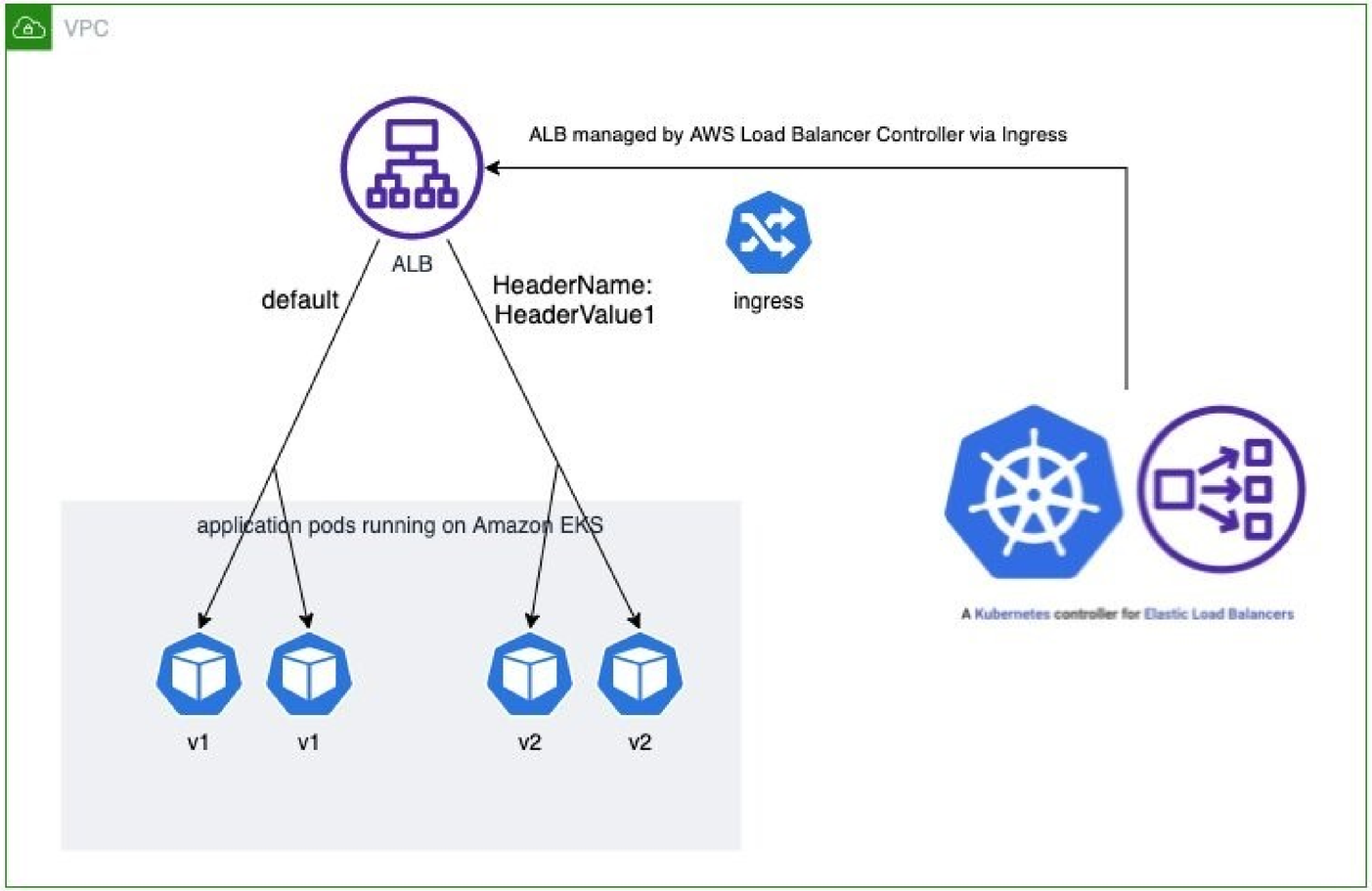
□□□□ - □□

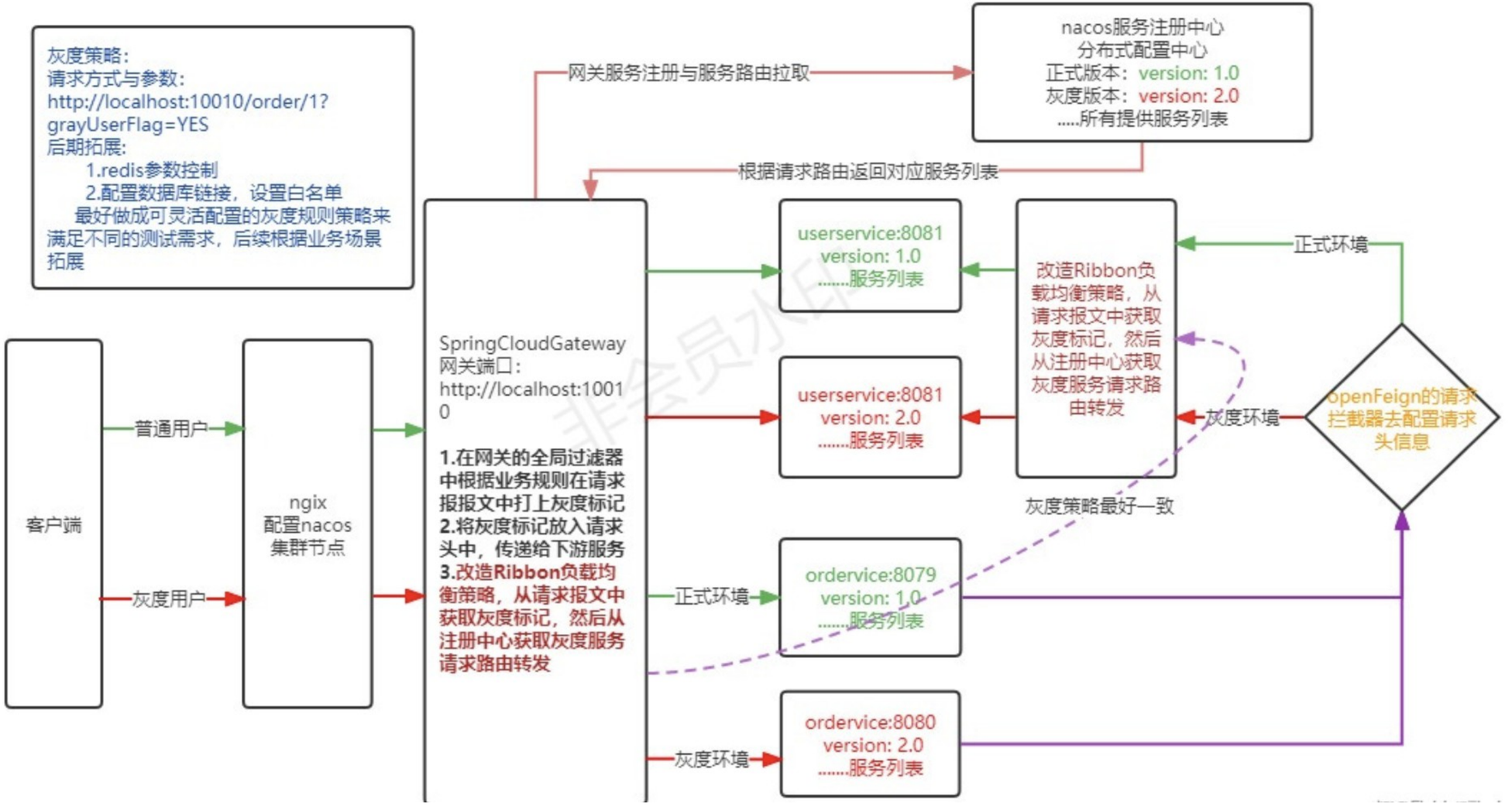
```
annotations:
  ...
 alb.ingress.kubernetes.io/actions.blue-green: |
  {
    "type":"forward",
    "forwardConfig":{
      "targetGroups":[
        {
          "serviceName":"hello-kubernetes-v1",
          "servicePort":"80",
          "weight":50
        },
        {
          "serviceName":"hello-kubernetes-v2",
          "servicePort":"80",
          "weight":50
        }
      ]
    }
  }
```

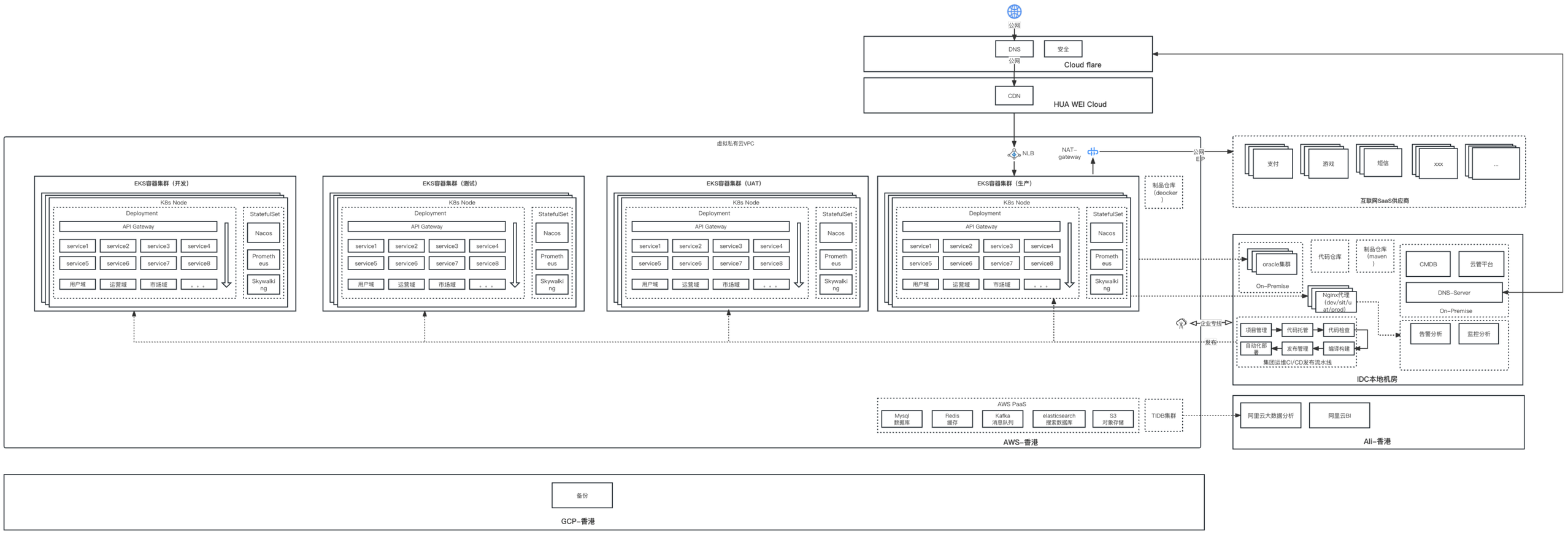


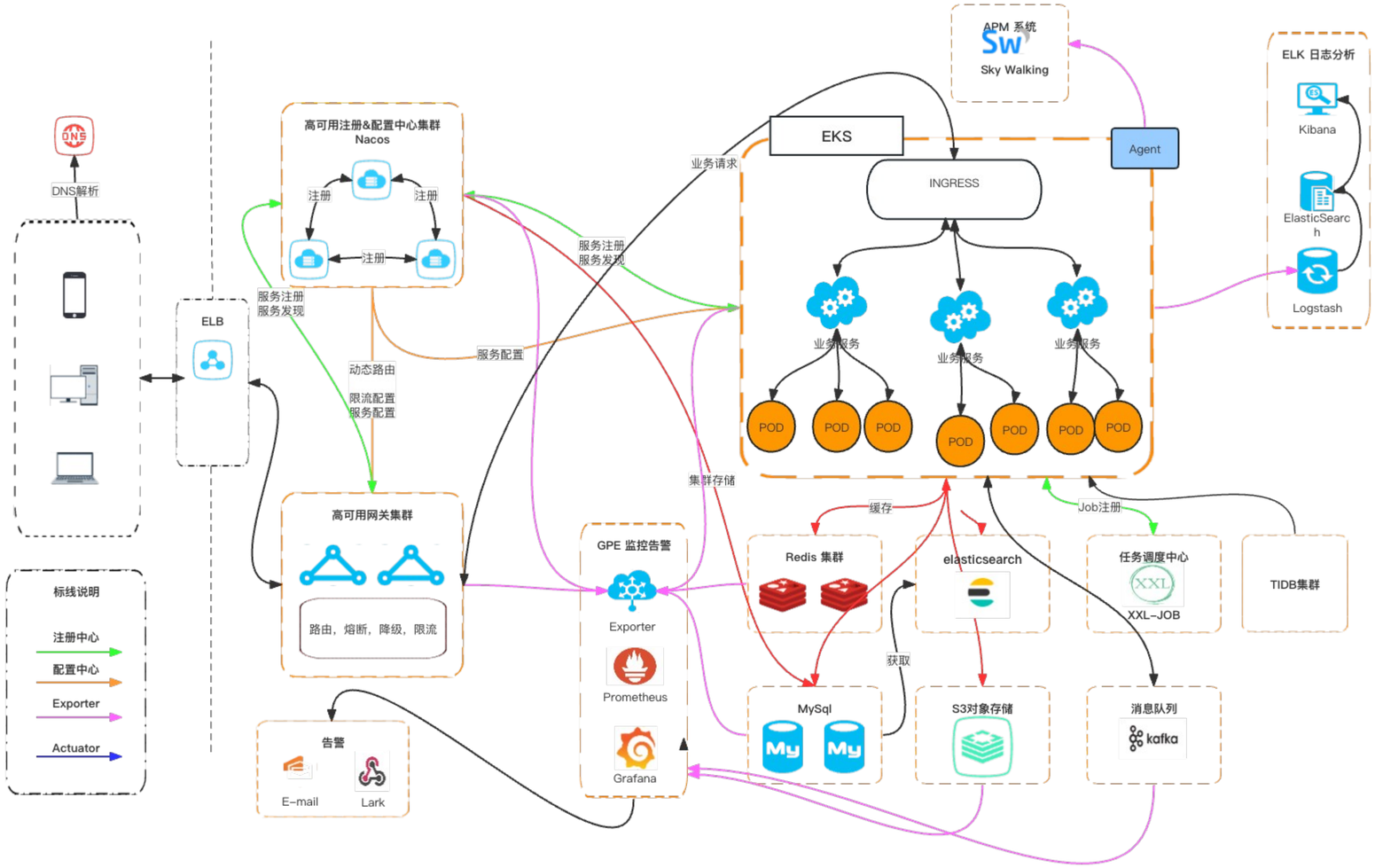
□□□□ - A/B test

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: "hello-kubernetes"
  namespace: "hello-kubernetes"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/conditions.ab-testing: >
      [{"field":"http-header","httpHeaderConfig":
          {"httpHeaderName": "HeaderName", "values":["HeaderV
      }]
    alb.ingress.kubernetes.io/actions.ab-testing: >
      {"type":"forward","forwardConfig":{"targetGroups":
          [{"serviceName":"hello-kubernetes-v2","servicePort":80}
      }}
  labels:
    app: hello-kubernetes
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: ab-testing
                port:
                  name: use-annotation
          - path: /
            pathType: Prefix
            backend:
              service:
                name: hello-kubernetes-v1
                port:
                  name: http
```




灰度策略：
请求方式与参数：
http://localhost:10010/order/1?grayUserFlag=YES
后期拓展：
1.redis参数控制
2.配置数据库链接，设置白名单
最好做成可灵活配置的灰度规则策略来满足不同的测试需求，后续根据业务场景拓展
网关服务注册与服务路由拉取
nacos服务注册中心
分布式配置中心
正式版本：version: 1.0
灰度版本：version: 2.0
.....所有提供服务列表
根据请求路由返回对应服务列表
客户端
普通用户
灰度用户
ngix
配置nacos
集群节点
SpringCloudGateway
网关端口：
http://localhost:10010
1.在网关的全局过滤器中根据业务规则在请求报报文中打上灰度标记
2.将灰度标记放入请求头中，传递给下游服务
3.改造Ribbon负载均衡策略，从请求报文中获取灰度标记，然后从注册中心获取灰度服务请求路由转发
userservice:8081
version: 1.0
.....服务列表
userservice:8081
version: 2.0
.....服务列表
改造Ribbon负载均衡策略，从请求报文中获取灰度标记，然后从注册中心获取灰度服务请求路由转发
正式环境
灰度环境
openFeign的请求拦截器去配置请求头信息
灰度策略最好一致
正式环境
ordervice:8079
version: 1.0
.....服务列表
灰度环境
ordervice:8080
version: 2.0
.....服务列表


公网
DNS
安全
公网
Cloud flare
CDN
HUA WEI Cloud
虚拟私有云VPC
NLB
NAT-gateway
公网EIP
EKS容器集群（开发）
EKS容器集群（测试）
EKS容器集群（UAT）
EKS容器集群（生产）
K8s Node
Deployment
API Gateway
service1
service2
service3
service4
service5
service6
service7
service8
用户域
运营域
市场域
。。。
StatefulSet
Nacos
Prometheus
Skywalking
制品仓库（deocker）
支付
游戏
短信
xxx
...
互联网SaaS供应商
oracle集群
On-Premise
代码仓库
制品仓库（maven）
CMDB
云管平台
DNS-Server
On-Premise
Nginx代理（dev/sit/uat/prod）
企业专线
项目管理
代码托管
代码检查
自动化部署
发布管理
编译构建
集团运维CI/CD发布流水线
告警分析
监控分析
IDC本地机房
发布
AWS PaaS
Mysql 数据库
Redis 缓存
Kafka 消息队列
elasticsearch 搜索数据库
S3 对象存储
TIDB集群
AWS-香港
阿里云大数据分析
阿里云BI
Ali-香港
备份
GCP-香港



Devops工具CODING

GitLab Jenkins docker HARBOR kubernetes RANCHER

测试管理

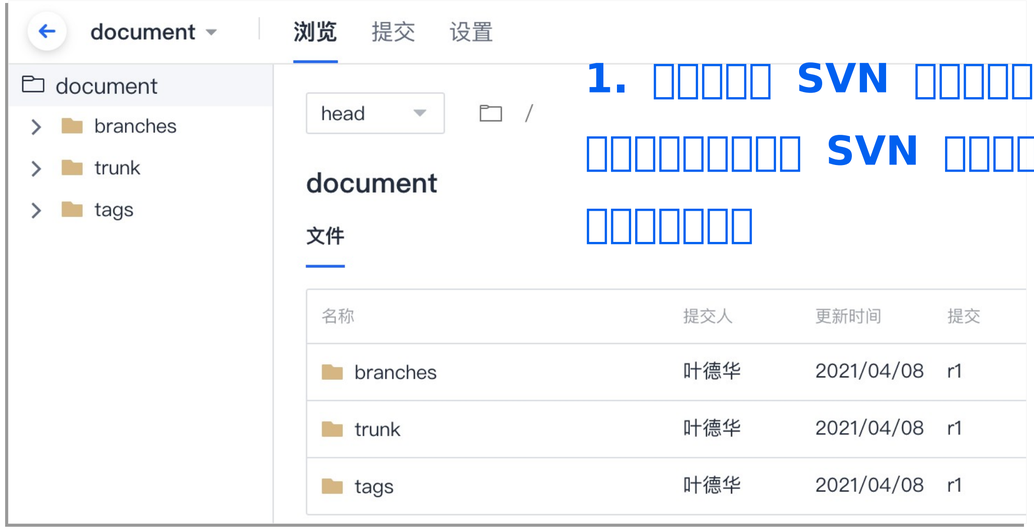
1. 组织和管理测试用例，支持在线编写、导
入

2. 灵活制定测试计划，探索、验证、回归……

3. 多人协同执行测试任务，测试缺陷快速与功能进行关联，开发人员及时修复缺陷

4. 测试结果可视化展现，自定义报告模板，定制生成测试报告通知指定负责人

[illegible]



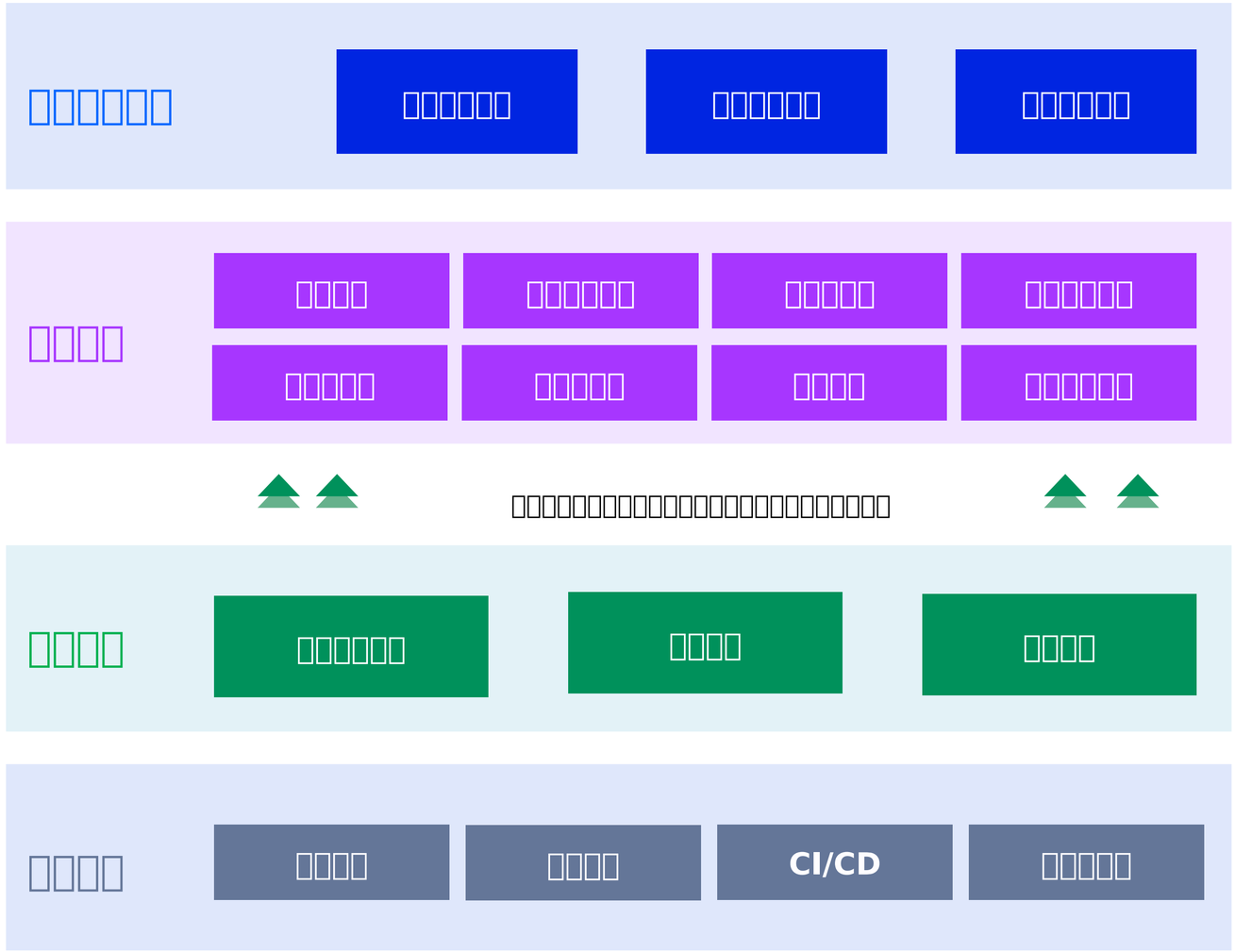
1. [illegible] SVN [illegible] SVN [illegible]

2. [illegible] Wiki [illegible] Markdown [illegible]

3. [illegible] Word [illegible] PPT [illegible] Excel [illegible] PDF [illegible] [illegible] SVN [illegible]

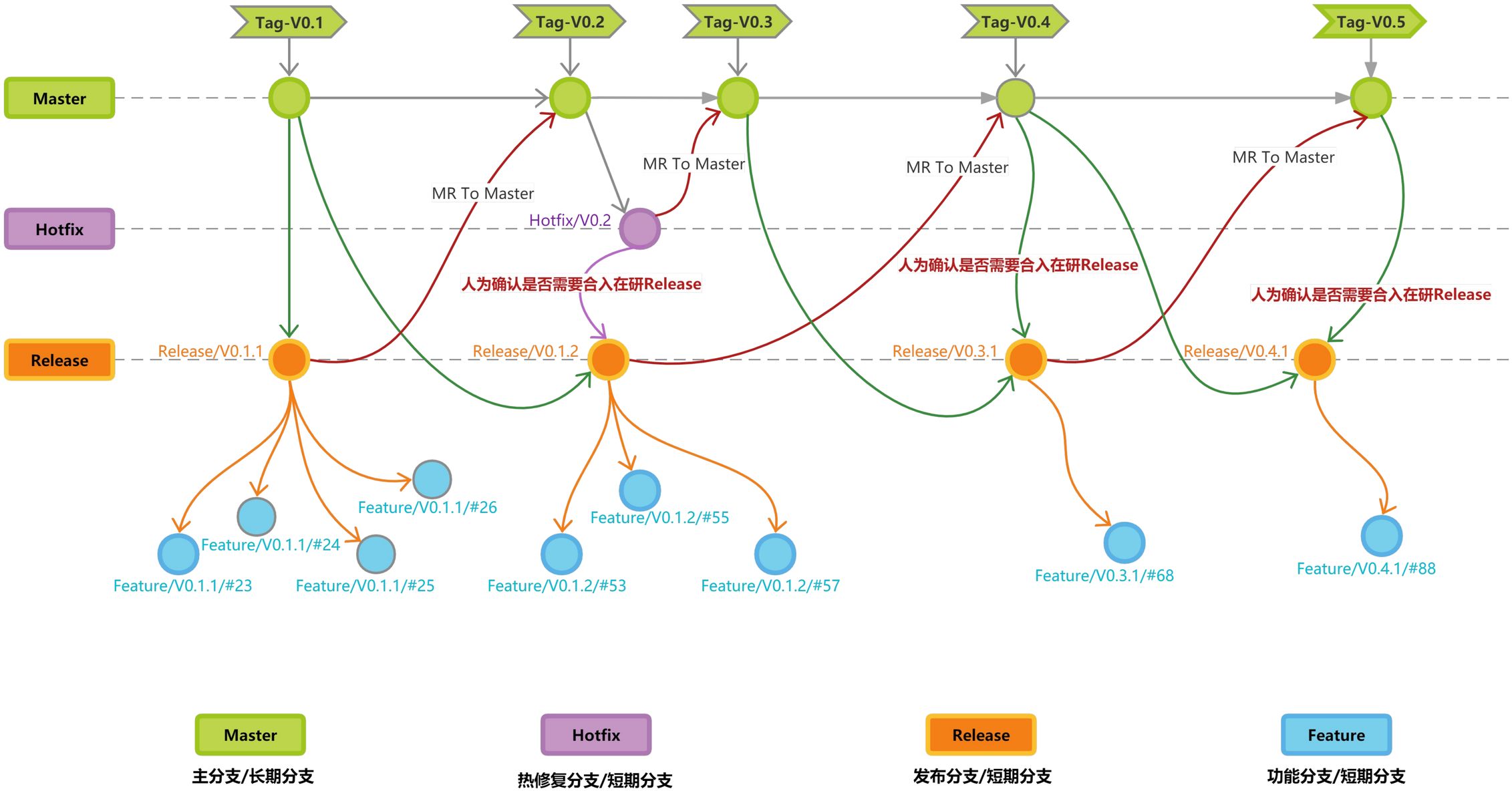
4. API [illegible] API [illegible] [illegible] Mock [illegible]

DevOps □□□



□□ □□□□□□□□ + □□□□□□□□□□□□□□□□□□□ DevOps □□□□□

□□□□□□□□□□□□



Master

主分支/长期分支

Hotfix

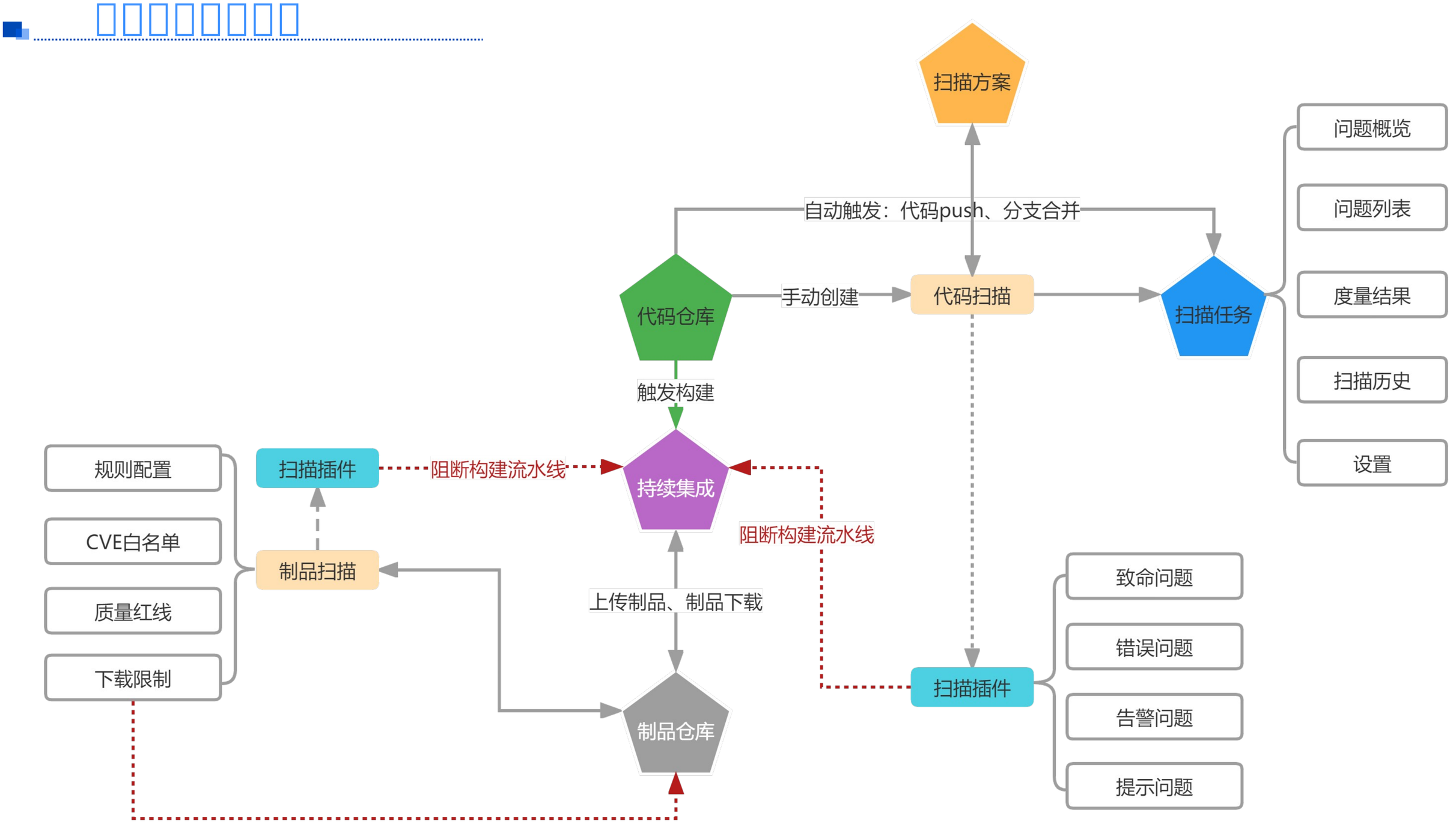
热修复分支/短期分支

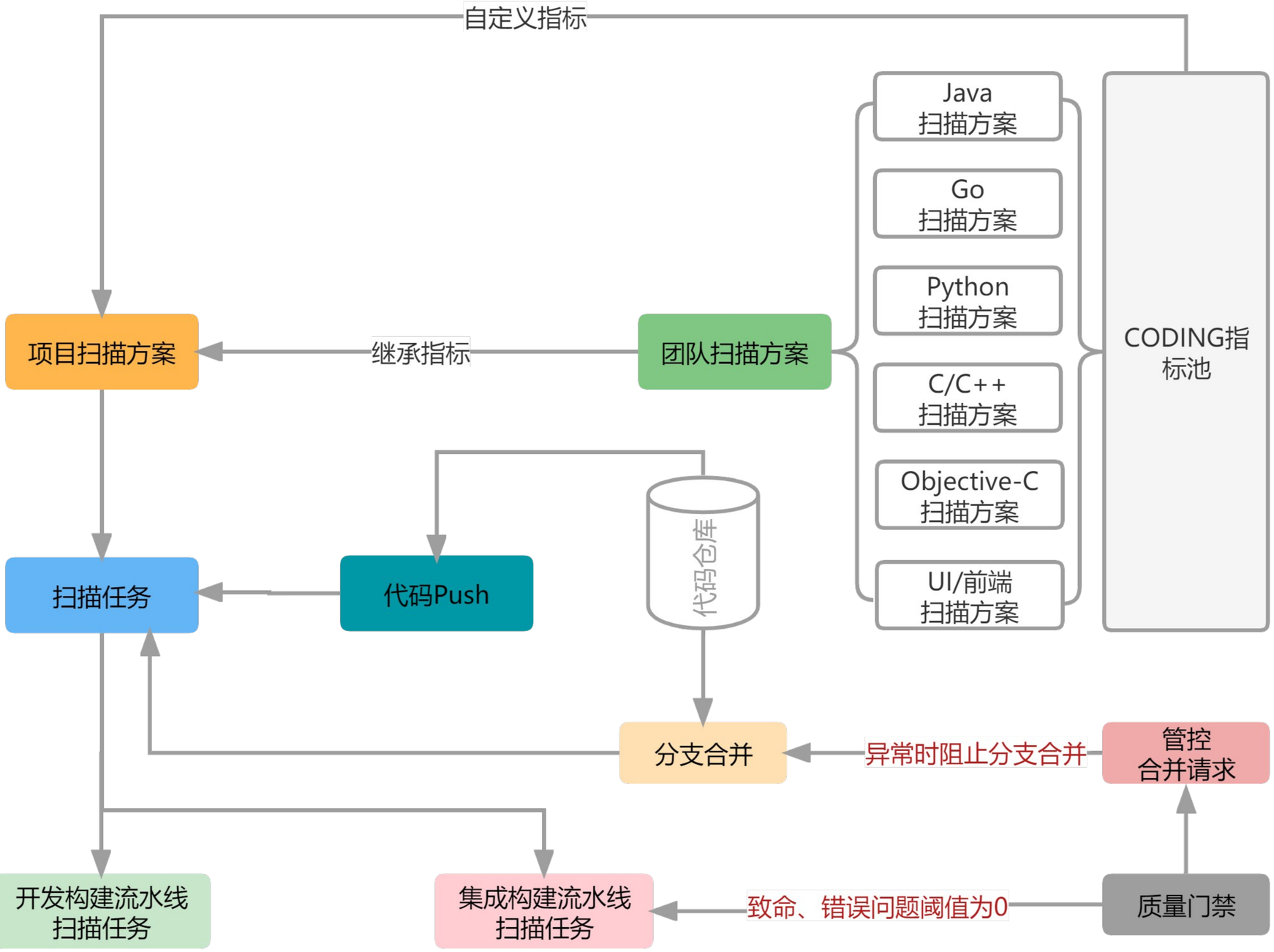
Release

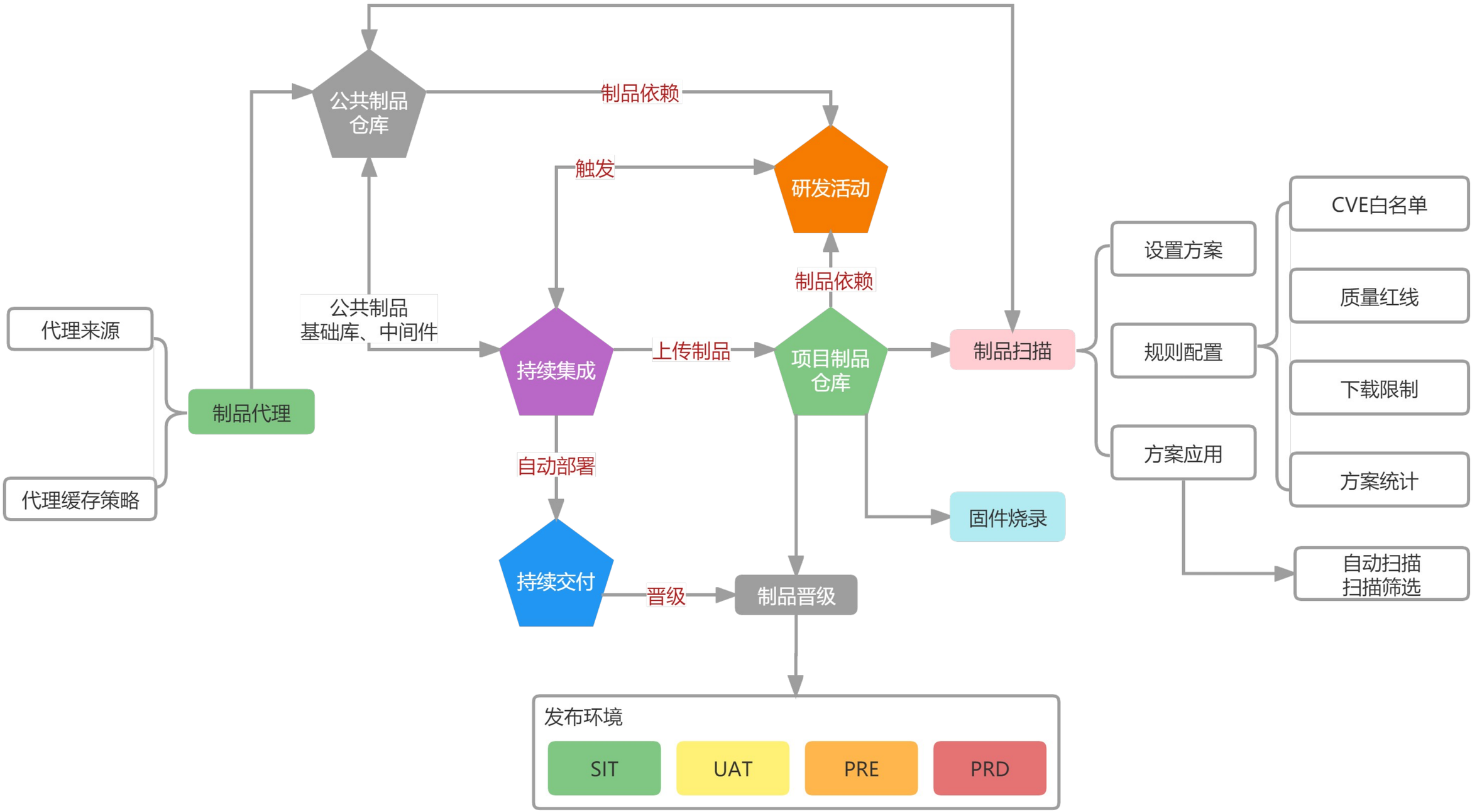
发布分支/短期分支

Feature

功能分支/短期分支


扫描方案
自动触发：代码push、分支合并
代码仓库
手动创建
代码扫描
扫描任务
问题概览
问题列表
度量结果
扫描历史
设置
触发构建
规则配置
扫描插件
阻断构建流水线
持续集成
阻断构建流水线
CVE白名单
制品扫描
上传制品、制品下载
质量红线
下载限制
制品仓库
扫描插件
致命问题
错误问题
告警问题
提示问题


自定义指标
项目扫描方案
继承指标
团队扫描方案
Java
扫描方案
Go
扫描方案
Python
扫描方案
C/C++
扫描方案
Objective-C
扫描方案
UI/前端
扫描方案
CODING指
标池
扫描任务
代码Push
代码仓库
分支合并
异常时阻止分支合并
管控
合并请求
开发构建流水线
扫描任务
集成构建流水线
扫描任务
致命、错误问题阈值为0
质量门禁


公共制品
仓库
制品依赖
触发
研发活动
制品依赖
公共制品
基础库、中间件
代理来源
代理缓存策略
制品代理
持续集成
上传制品
项目制品
仓库
制品扫描
设置方案
规则配置
方案应用
CVE白名单
质量红线
下载限制
方案统计
自动扫描
扫描筛选
自动部署
固件烧录
持续交付
晋级
制品晋级
发布环境
SIT
UAT
PRE
PRD

□□□□